- In lab1, I mainly implemented Tuple, HeapFile, Catalog, BufferPool, and SeqScan Operator. For each Tuple object, it has a TupleDesc object storing the type and (optional) name for each column of that tuple. Then, HeapPage is implemented on top of Tuple, representing a page of HeapFile (a table) and containing a bitmap and a list of tuples. The bitmap is used to identify empty slots in a HeapPage. Also, within a HeapPage, there is a HeapPageId, a unique identifier for each page so that the buffer pool can quickly know which pages it caches. For each Tuple in a HeapPage, it also has a RecordId to locate the tuple's position on the page, thus the disk. Catalog is implemented to keep track of the metadata of tables in the database and is referenced by Database. Everytime I want to access some metadata of a table, I should call Database.getCatalog() to make sure that there is only one global instance of Catalog. I also implemented BufferPool, which caches Pages that are recently read. getPage() method checks if the page is in the cache. If not, it will ask the HeapFile access method to read the page from the disk. Whenever an operator needs to access a page, it should call getPage() method in BufferPool instead of readPage() method in HeapFile. I also implemented HeapFile, which represents a table in the database and consists of a collection of HeapPages. Each HeapFile has a unique identifier which can be accessed by calling getId(). HeapFile is also used to read/write pages from/to the disk. Also, HeapFile has an iterator for iterating over all tuples in that HeapFile, which is implemented on top of iterators for each HeapPage. The HeapFile iterator loads in the first HeapPage of that file when open() is called, and the iterator starts working. Finally, I implement the SeqScan Operator, which sequentially returns all tuples in the HeapFile specified with tableid.
- When implementing the iterator in HeapFile, I added 2 public fields in AbstractDbFileIterator to keep track of which page the iterator is on, and that HeapPage's iterator.
  In Catalog, I used two maps, one mapping table id to table name and the other mapping table name to (DbFile, primary key).
  In readPage() in HeapFile, I throw all exceptions when reading a page on disk.
- Add a test for illegal pid when calling readPage() in HeapFile. The test first creates a 1-page HeapFile, calls readPage on the second page, and checks if an exception is thrown.
- No API changed
- All parts completed